Lab 4: Speed Testing

Import the project to Eclipse
File -> import projects from file system or Archive -> select directory for import source

Add Junit 4 If needed.
Right click java project -> Build Path -> add Libraries -> select Junit 4 -> click finish

**** Only edit the //TODO section in the code**

1- Understanding the Account and Customer Class

2- Write tests for account Class

Write tests in **//TODO** section below Customer class in HandsOnExpereince.java and run the test case. The test depositIncreasesBalance is a general statement about the behavior which you are trying to verify. E.g: Assert that the balance after depositing is greater than zero.

Write two tests:

```
@Test
  public void hasPositiveBalance() {
    account.deposit(50);
    assertTrue(account.hasPositiveBalance());
  }

@Test
  public void depositIncreasesBalance() {
        int initialBalance = account.getBalance();
        account.deposit(100);
        assertTrue(account.getBalance() > initialBalance);
  }
```

Question 1: Do the test cases above successfully run ? Why or why not?
(Provide two screenshots of these tests and justify the result for each of them)

3- Initialize Account instance

Add a @Before method

```
@Before
  public void createAccount() {
        account = new Account("CompanyName");
  }
```

Question 2: Do the test cases successfully run ? Why or why not?
(Provide two screenshots of these tests and justify the result for each of them)

4- Write a test case with asset explanation

The goal is to write a test case with an explanation and run the test case.)

Inspect and run the following test case:

```
@Test
public void testWithWorthlessAssertionComment() {
    account.deposit(50);
    assertThat("account balnace is 100",account.getBalance(), equalTo(50));
}
```

Question 3: Does the test above run or not, please explain and provide a screenshot?

Question 4: Write a new test case to show some potential error in the code, and submit the test (code and expected result) and a screenshot of the result. Your answer should make the test descriptive (showing a potential weakness and how you address it).

5- Using @Test annotation

The goal is to verify that exceptions get thrown when expected because understanding the conditions which cause a class to throw exceptions will be easier for the user. The @Test annotation supports passing an argument which specifies the type of an expected exception. It is advisable to not clutter tests with try/catch blocks to deal with checked exception. Instead, rethrow any exceptions from the test itself.

```
@Test(expected = InsufficientFundsException.class)
    public void throwsWhenWithdrawingTooMuch() {
        account.withdraw(100);
    }
```

Question 5: What happens if the @Test annotation without specifying the exception?

```
@Test
    public void throwsWhenWithdrawingTooMuch() {
        account.withdraw(100);
    }
```

Provide screenshots for both cases (with and without specifying the exception) and comment on the difference between them.